Requirements for Job Monitoring

Adam Lyon, ...
22 September 2010

In the course of discovering requirements for Grid Monitoring, it is important to think about what users and operators do when they run jobs or monitor the system. Let's do this without worrying about implementation, or even if an implementation is possible.

1 Use cases for Users:

Before a user submits a set of jobs to the system, there are several things they want to discover:

- Is the system working?
- How busy is the system? and related, About how long will it take before my jobs run?

The answers to these questions go into the decision about whether to submit jobs at that time (e.g. if the system is broken, don't bother) and how many jobs to submit (e.g. if the system is really busy, maybe submit the most important jobs first; if the system is lightly used, maybe submit more jobs than would otherwise).

Once the jobs are submitted to the system, the user then wants to know and track the following:

- What state are my jobs in? (Queued, Running, Completed, Failed, Mysteriously disappeared?). Why have queued jobs not gone running? (Where is my place in the queue or fairshare?)
- For the jobs that are running, are they progressing normally? If not, then why? (Are they stuck waiting for data? Are they churning the CPU without advancing to the next file in a timely manner?)
- Did jobs that recently complete do so successfully? Did the output data get stored back? How long did the jobs run for? How much data did they read? Write? If they failed, why (Crash? Grid problem? Did the job violate site rules?)
- Users may also want summary information of the jobs they submitted together (e.g. e-mailed to them when the set of jobs complete).

Furthermore, it is also useful for users to look at other people's jobs:

- Is everyone's job at this site working or having problems like mine?
- Is a different user hogging resources at a site?
- Is a different experiment hogging resources at a site?

2 Use cases for Operators:

Here, operators are *not* site-admins nor system-admins. Rather they are personnel who monitor

and troubleshoot aggregate data handling and job handling systems for the experiments.
The basic job of an operator is to determine whether or not a system is working properly and troubleshoot if not. Furthermore, operations management may also like to know the usage level of a system by the experiments (such people are included here under "operators"). A list of things that operators need to use monitoring to discover are,

- Is the system working? Are jobs running successfully? How many are stalled? Is data flowing to jobs in a timely manner? For certain jobs that have problems, it may be important to view system information for the job's machine (perhaps virtual machine); e.g. load average, iowait, ...
- If jobs are failing.... How many jobs (users) are impacted? What is the distribution of errors (is there one error everyone is getting or are there lots of different errors). Are the errors happening at one site or multiple sites? Are errors occurring at the same time?
- If data handling is failing... [Same questions as above]
- Are resources being utilized effectively? Are lots of slots free? Are lots of slots running but using zero cpu?
- Are some users hogging resources?
- What is the distribution of resource utilization by experiment?
- Operations will also want to see much of this information historically.

3 Requirements

Below are requirements for a job monitoring system, given the use cases above.

3.1 Overall system monitoring

Monitoring should indicate, for each grid site accessible to the experiment(s), the following:

1. How many slots are available to the experiment (this may be impossible to determine for sites used opportunistically)
2. How many slots are in use by the experiment
3. How many jobs are queued
4. How many jobs are "stuck" (that is jobs that are in a running state, but using no CPU)
5. How many jobs have low efficiency (that is jobs that have taken long wall clock time but little CPU time)
6. How many jobs have completed within some time interval (there should be one or several default time intervals (e.g. 3 hrs, 6 hrs, 24 hrs, 72 hrs) and the time interval should be settable by the viewer
7. Like 6, but completed jobs subdivided by the exit code
8. Like 6, but completed jobs subdivided by efficiency bin (cpu time / wall clock time)
9. Items 1-8 viewable as a strip-chart histories with default and viewer-settable time interval (e.g. view history of # of open slots, history of # of running jobs) -- perhaps some strips are superimposed for comparison
10. Items 1-8 sub-divided / selectable by user
11. Items 1-8 aggregated over all sites
12. Items 1-8 aggregated over all sites and sub-divided / selectable by user

3.2 Overall data handling monitoring

1. File delivery statistics for each site, within a specified time period
    a. Scale of file deliveries to site (# files, TB, events)
    b. Average file delivery time
    c. Outlier delivery times
2. For each site, list of file delivery errors for a specified time period with detailed error information

3.3 Job level monitoring

Here, we assume the job runs within a VM (for the sites where jobs run on the bare metal, then quantities below will be for the whole machine).

In association with item 3.1 above, there needs to be a way to select job(s) to view. Users should be able to select jobs by looking at the list of jobs they submitted to the system, subdivided by state. Furthermore, operators should be able to select jobs by listing jobs that match criteria (e.g. jobs with low efficiency that have completed today; jobs from user A running on site B that have been stuck for more than n minutes; jobs on machines with high iowait).

For each job selected to view...

1. List basic information (job owner, job ID(s), site, state [queued, running, stuck, completed], time in that state
2. If the job is complete, then the completion information (exit status, error messages)
3. A log of the various state transitions (e.g. when did the job go from queued to running? How long was it in each state?) The log should also indicate how long the job was waiting for each file and busy working on each file
4. Log of files that were read in (name, when, size, source)
5. Log of files that were written out (name, when, size, destination)
6. If waiting on a file, how long has it been waiting
7. If working on a file, what file is being read (name, size, source)
8. VM information (load average, CPU time, iowait, memory usage, disk space) as a history plot over the life of the job
9. Indication of the job efficiency
10. Access to log files the job is producing while it is running

The monitoring system should also alert users (via e-mail or some other communication mechanism) when a set of jobs have completed (e.g. jobs from the same submission). The message should contain basic statistics on the job (e.g. exit status, efficiency, ...).